Project Integration Architecture: Formulation of Dimensionality in Semantic Parameters

Dr. William Henry Jones
National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, OH 44135
216-433-5862
William.H.Jones@grc.nasa.gov

X00.00 24 Mar 2000
X00.01 19 Sep 2000

Keywords:
PIA; PRICE; CORBA; Knowledge Management; Application Integration;
Technical Integration Technologies; Semantic Encapsulation;
Dimensional Encapsulation; Self-Revelation;

ABSTRACT: *One of several key elements of the Project Integration Architecture (PIA) is the formulation of parameter objects which convey meaningful semantic information. The infusion of measurement dimensionality into such objects is an important part of that effort since it promises to automate the conversion of units between cooperating applications and, thereby, eliminate the mistakes that have occassionally beset other systems of information transport. This paper discusses the conceptualization of dimensionality developed as a result of that effort.*

1 Introduction

The analysis of the whole of an engineering system frequently involves a number of cooperating analyses, each focusing on a particular discipline of analysis relevant to the whole. As discussed in [1], one effort of the Project Integration Architecture (PIA) [2] has been to define and develop semantically meaningful parameter objects so that the nature of the information provided by a parameter may be usefully determined by automated examination of the object encapsulating that parameter. By so doing, it is expected that the cooperating analyses may effectively transfer information by the inspection of each other's parameter objects.

The parameters of analyses are often dimensional in nature. For instance, the span of a turbine blade is given as so many inches or the thrust of a rocket as so many Newtons. The recognition and encapsulation of this dimensionality is considered a key step in the process of semantic parameter definition and implementation.

Dimensionality involves two elements: the form of the dimension (length, mass, velocity, and the like) and the system of measurement within which the a value is stated (force as stated in English pounds as opposed to metric Newtons). Information characterizing these two aspects may be combined to provide a correct interpretation (in so far as a numeric value is concerned) of any given value.

The original implementation of these two concepts for the PIA effort was simple and straight forward: a code value was recorded specifying the system of measurement and the object kind mechanism identified the form of measurement. For example, a scalar length object existed and it, when interrogated, would yield the code number of the system of measurement within which its value was relevant. This simple approach proved satisfactory for simple inspection and transfer of the encapsulated information; however, the logical extension of objects based upon this foundation demonstrated an operational problem which proved this simple formulation less than completely satisfactory.

2 Particulars of the Problem

The PIA implementation developed a series of dimensional objects capturing dimensional form and system of mea-

surement as simple concepts. Thus, a series of scalar, vector, and matrix forms for length, mass, velocity, and the like were developed. Each such object kind would yield a code value indicating the system of measurement within which its encapsulated value existed. Codes for metric and English systems were defined. Further, each object kind would identify a shared table of conversion factors appropriate to that kind. Thus, length objects identified a table of conversion factors for length, mass objects a table for masses, and so on.

This set of developed objects was then used as the foundation for a further series of objects encapsulating the semantics of geometry as obtained from typical Computer Aided Design (CAD) systems. As such, this geometric object set focused largely on points (in three-dimensional space), collections of points, vectors (again, in three-dimensional space), groupings of points into triangles, groupings of triangles into face tesselations, and the like. The various length object forms provided a very natural base upon which to build these geometric entities and, to this point, all proved well.

One of the needs that arose as work progressed was to extract cross-sectional area from the geometric entity described by the aggregation of data. (In fact, the focused activity involved an air-breathing propulsion system inlet which was not axi-symmetric. The inlet system was, of course, to be analyzed by a one-dimensional code which, naturally, presumed axi-symmetry in computing its flow area values from centerbody and cowl radii.) Having just formulated vectors, unit vectors, unit surface normals, and the like for the purposes of defining geometric characteristics, it seemed very natural to use these tools in the solution of the cross-sectional area requirement.

2.1 The Train Wreck

The definition of geometric planes and the computation of intersections with such planes (as well as many other tasks) is well suited to vector manipulations. Cross products (outer products), dot products (inner products), vector differences, and other such manipulations can quickly solve such problems as locating the point at which a given line segment intersects a defined plane in space.

But it was with these very vector calculations that the inadequacy of the dimensional formulation was found. A cross product of two length vectors yields not a vector with units of length, but a vector with units of area. A further cross product of that result with a third length vector now yields a vector with units of volume. Similarly, dot products yield areas, not lengths.

The first glaring difficulty was that the dimensional objects were inflexible. A length object explicitly declared itself to be in units of length; feet, meters, what have you. It could not account for a cross product being in units of length squared. Clearly, the cross product of two length vector objects could not be encapsulated in a third length vector object because its units would not match.

The next logical step was to find some object kind that could encapsulate the cross product of two length vector objects. Since the units of the result were length squared, the eye first cast its attention upon the area vector object form. Regretably, that proposition did not last long.

The first difficulty with encapsulating a cross product of lengths into an area object was the simple question regarding the precise nature of a vector of areas. A cross product of two length vectors does indeed have an interpretation as a vectorialized area (that is, as a single area having magnitude and direction); however, if such a result were encapsulated into an area object, it would be indistinguishable from a vector having three area components. One of the tenants of semantically meaningful object derivation is that the encapsulated information should make sense, but a vector of areas does not make such sense unless the further leap of logical rearrangement from vector of areas to vectorized area is made.

The next difficulty was to identify the logical extension of the area object answer when a second cross product was to be taken with the results of a previous cross product operation. Now the units of the result would be length cubed which would no longer fit in an area object. Objects with volume dimensionality existed, but a vector of volumes makes even less sense than a vector of areas. Further, the volume object answer demonstrated the difficulty of extension; there was no length-dimensional form beyond volume, so a further cross product of cross products would require invention into the great beyond.

Another problem was what to do with a unit vector object, an accomodating specialization of the vector object which conveniently normalizes itself. If it resided in a length vector object, it would convert its values between measurement systems. Oddly, a unit vector one foot long in English units would be about one third of a meter long in metric units. This behavior seemed antithetical to the defined nature of the object and was considered not at all satisfactory.

As if these problems were not enough, operational problems existed. Code which constructed a cross product would have to know or determine the kind of object in

which to encapsulate the result. It would have to look at the cross product of two length vectors and know to produce an area vector. Not only would it have to know to produce an area vector, but it would further have to have a method to determine what kind of object having area characteristics (since the basic area object kind is, again, only a base upon which further derivation is to occur) was the correct kind of object.

Beyond this, assuming a scheme for identifying the encapsulating object kind could be devised, the next difficutly was to keep the ability to form a cross product associated with the results of that action. A cross product functionality was a very natural addition to a length vector object, but it seems an unnatural ability for objects of area, volume, and beyond. It is, further, seemingly unnatural to make a cross product functionality that accepts areas and volumes as its input. While such can be coded, it seems a very odd thing.

While thrashing about with all of this, yet another question came up: what was to happen when one wanted to divide a length by a time and come up with a velocity? Again, there would be a search for an object. But what object?

All in all, the situation was beginning to look very muddled, indeed. Not at all the sort of thing one wishes to let loose upon an unsuspecting world.

3 Solution: Save the Train, Wreck the Brain

Faced with the difficulties described in the preceding section, the need for a distinct reconsideration of the formulation of dimensionality was clear. The key was found in the need to keep such functionality, in this case the ability to compute a cross product, associated with the results of the operation. A cross product of two vectors is still a vector which often is involved in further cross product operations (or dot products, or vector magnitudes, etc.). Thus, it was realized that a cross product was most appropriately encapsulated in a length vector object where the functionality to compute cross products and other vector operations resided.

Given this new choice, it was then apparent that a length object could not be inflexibily declared to be in units of length. This seemed to provide one of two choices: either dimensionality could not be meaningfully encapsulated in objects or the meaning of dimensionality had to be altered into a more flexible formulation. Since the first choice was antithetical to the presumptions of the project, it was determined that dimensionality would take on a more flexible form.

3.1 Flexibility Through Separation

The desired dimensional flexibility was found in a simple separation. Dimensionality was separated into two components: a characteristic and an application. The characteristic is the mixing of fundamental dimensional elements into a composite dimensional form. For instance, the characteristic of velocity is length divided by time. The application is the extent to which that characteristic is applied in a particular instance of the dimensionality. In the case of the cross product of two length vectors, the characteristic remains length while the application of that characteristic doubles from one to two.

This new view of dimensionality leads to the somewhat mind-bending (if not altogether mind-breaking) view that a length object, while being fundamentally a length object, may be, in fact, measured in length squared or length cubed, or whatever. It also leads to the question as to just how length squared is different from area, the answer being rather enigmatically that length squared is not entirely the same thing as squared length.

Despite this initial difficulty of conception, following the fingers across the keyboard for a while does demonstrate a certain rationality to the idea. Consider first the case of the troublesome unit vector. It achieves unity by obtaining its magnitude (the square root of the sum of the squares) and dividing each element by that value. Now, without regard to whether the vector was originally measured in units of length or length squared or length to the anything else, it is quite obvious that the normalization operation reduces the vector to units of length to the zero power. That is, a unit vector has the characteristic of length, but an application of that characteristic of zero. This, in turn, makes the unit vector non-dimensional and, as a further consequence, invariant between systems of measurement. Now, suddenly, the unit vector has a magnitude of unity without regard to the system of measurement in which it is viewed. It will be one meter long in metric and one foot long in English. Curiously, the very property that was lacking before.

3.2 Dimensional Recombination

As illustrated above, the rules of mathematical combination for values of dimensional nature are those commonly understood in conventional manipulations. Addition and subtraction may only be performed between values in which the aggregate of dimensional characteristic and application are identical, sometimes referred to within the PIA environment as aggregate dimensional congruence.

Multiplication and division can be performed between dimensional quantities of any form and results in a computed dimensionality in which the power of application is, perforce, unity. In order to assure that the dimensional result is neither corrupted nor inappropriately encapsulated, these operations are currently implemented to place the result in dimensionally polymorphous base class objects. Usually, these objects are treated as temporaries and their contents assigned to objects of the correct dimensional semantics.

3.3 Encapsulation of Results

The problem of encapsulating a result of computed dimensionality is solved by a minor jog in assignment protocols. There are two phases of mixed-dimensional computational result encapsulation: capture and assignment. That is, when one writes the statement *a = b * c*, there are two acts: capturing the result of *b * c* and then assigning that result to *a*.

The first act, result capture, needs only to preserve the result value and its dimensionality correctly. As stated above, those operations resulting in such computed dimensionality place their results in dimensionally polymorphous base class objects of the correct structural kind. The base form, while not knowing of the various derived dimensional characteristics, still understands fully the nature and implementation of computable dimensionality and, thus, is entirely suitable for the temporary capture of a computed, mixed-dimensionality result.

The second act, that of assignment, is half solved by the act of coding. Typically, the nature of the expected result is known: one divides a length by a time expecting a velocity. The programmer, in writing *a = b / c*, will thus make *a* a velocity object. The difficulty arises in that normal object assignment protocol (at least as the PIA project defines it) requires assignment to an object to be from an object of the destination object's kind. While a velocity may be a kind of dimensional object, a base-class dimensional object is not a kind of velocity and, thus, normal protocols prohibit the assignment from a temporary, dimensional base object.

The solution to this problem is, of course, quite clear: the object kind restriction in assignment is relaxed in the case of dimensional objects. This is implemented through an override of the inherited assignment permission function and assignment from a base class to a derived class becomes possible. In doing this, though, all sheets are not simply cast to the wind. First, the source object is required to be a dimensional object of the same structural nature. That is, a vector must be the source of a vector assignment, a scalar the source of a scalar assignment, and so on.

Beyond the basic assignment requirements, the following cases are then treated. Note that the first case treats all assignments to a dimensionally polymorphous base class. Thus, the other cases are all understood to be assignments to derived classes whose dimensional characteristic is established.

1. Assignment to a dimensional base class from any dimensional class is always allowed. In this case, the dimensionality of the source object is copied without adjustment resulting in pure dimensional congruence.

2. If the assignment is from a dimensionally polymorphous base class and the dimensionality of the source is similar in its dimensional characteristic, the assignment is allowed. The dimensional power of application of the target object is adjusted as necessary to bring the source and target objects into aggregate dimensional congruence.

   The similarity of the dimensional characteristic is established through computation with an anticipation of application adjustment. For example, a characteristic of length squared with an application of unity is considered similar to a characteristic of length with an application of two and will allow assignment to go forward.

3. If the assignment is from a dimensional object, the assignment is allowed if the source and target are congruent in their dimensional characteristics. Again, the dimensional power of application of the target object is adjusted as necessary to bring the source and target objects into aggregate dimensional congruence.

The difference between the second and third cases is nearly invisible in this description; however, the difference does exist in implementation. As noted in the second case, similarity is established by computation with an anticipation of application adjustment. As with similar as opposed to congruent triangles, the computation establishes that the basic shape of the dimensional characteristics is the same. This allows the computed dimensionality of a temporary result (such as produced by the essential multiplication operation of a vector cross product) in which all the dimensionality has been placed in the characteristic (while the application has been set to unity) to be assigned back to its nominal class through an adjustment of the application factor. (An adjustment to the source dimensional characteristic is implied by this, but is not actually carried out since the assignment operations only amend the contents of the target operand. What does in fact happen is that the application

factor placed in the target object is that which would have resulted had the source been adjusted to obtain congruence of its dimensional characteristic.)

4 Summary

The conceptualization of dimensionality and the encapsulation of those developed concepts into dimensionally-aware objects has been discussed. The actual operations involved are far from revolutionary. Dimensional conversions and manipulations have long been well known and understood. On the other hand, the infusion of these simple operations into the automation of object programming at this junction appears to be a useful step forward. When the PIA effort is migrated to net-accessible distributed-object technology (an effort that is currently under way), it will be possible to access application information with the confidence that the dimensional nature and integrity of that information will be automatically preserved and accounted.

References

[1] William Henry Jones. Project Integration Architecture: Formulation of Semantic Parameters. Draft paper available on central PIA web site, January 2000.

[2] William Henry Jones. Project Integration Architecture: Application Architecture. Draft paper available on central PIA web site, March 1999.